Indian Statistical Institute
M.Tech.(CS) — First Year
Programming Test 2022
Date: 01.08.2022 Duration: 16:00–18:00

INSTRUCTIONS

1. Please make sure that your programs adhere **STRICTLY** to the specified input and output format. They should not require any input or produce any output beyond what is specified in the input and output format specifications below.

2. Each program should be saved in a file whose name has the following form:

   `<your roll number at ISI in lower case>-<problem number>.c`

   For example, a student having roll number CS2250 should save his/her solution to problem 1 in a file named `cs2250-1.c`.

   Please **STRICTLY** follow the above file naming conventions when saving / submitting your programs.

3. Please upload your programs to `https://www.dropbox.com/request/vCpxqSijh4eNC5uPbR1W` when you are done.

4. In order to pass, you will have to write correct programs for at least 2 out of the following 3 problems.

1. Write a C program that can add one positive integer to another positive integer, and subtract one positive integer from another positive integer. The input integers may contain up to 100 digits (0–9). Thus, you will not be able to use the usual builtin types (`int`, `long int`, etc.) to store the inputs. You may **not** use the GNU Multiple Precision (MP) Arithmetic Library, or any other similar library or C++ class.

   Input format: You will be given 3 strings via standard input (i.e., the terminal). The first two will be the two integers; each integer will consist of a sequence of up to 100 digits. The two integers may not necessarily contain the same number of digits. The third string will consist of a single character, either `+` or `-`.

   Output format: If the third string is `+`, your program should print the sum of the two numbers; if it is `-`, your program should print the result obtained by subtracting the second number from the first (in this case, the second number will be no greater than the first number). Note that the sum of two 100-digit numbers may contain more than 100 digits.

   Sample input 1

   `100...005` (50 digits in all)
   `100...005` (50 digits in all)
   `+`

   Sample output 1

   `200...010` (50 digits in all)

Sample input 2

```
100...010      (100 digits in all)
10...005       (99 digits in all)
-
```

Sample output 2

```
90...005       (99 digits in all)
```

2. Given the Roman alphabet $(a, b, c, \ldots, z, A, B, C, \ldots, Z)$, we define a *letter bi-gram* as any ordered pair of 2 letters from the alphabet. Thus, aa, ab, kZ are all valid letter bi-grams.

   Write a program that takes some text as input, and counts and prints the frequency of each letter bi-gram that occurs in the text. The input text may contain both uppercase and lowercase letters, spaces, newline characters, digits, punctuation marks, etc. You should ignore case when counting bi-grams. Thus, you should count AC, aC, Ac, and ac as four occurrences of ac.

   Input format: Text will be provided through standard input (i.e., the terminal). The input may span multiple lines. An empty line will mark the end of the input.

   Output format: Your program should print, in lexicographic order, the letter bi-grams that occur in the text, along with the number of occurrences of each bi-gram.

   Sample input 1

   ```
   aBc
   d,e
   ```

   ⟵ empty line to mark end of input

   Sample output 1

   ```
   ab 1
   bc 1
   ```

   Explanation: `B` is a part of two bi-grams, while `d` and `e` are not part of any bi-grams, since they are preceded by and/or followed by a non-letter.

3. Let A, B and C be 3 matrices with integer entries. Write a program to compute ABC. If ABC is not defined, your program should print an error message. The sizes of A, B and C will only be known at runtime. Your program should dynamically allocate memory to store these matrices. You should also try to minimise the number of arithmetic operations required to compute the product. You may assume that all entries of all the matrices involved can be stored as `int` type variables.

   Input format: The input, provided via the terminal, will consist of two positive integers m_A and n_A, corresponding to the number of rows and columns of A, followed by the $m_A \times n_A$ entries of A. Next, the dimensions and entries of B and C, respectively, will be provided using the same format as for A.

   Output format: If the matrix product ABC is not defined, your program should print `ERROR` on the screen and exit. If the matrix product ABC is defined, your program should print the entries of ABC in matrix form;

i.e., entries of the first row should be printed separated by spaces on the first line; thereafter, the entries of the second row should be printed on the second line, and so on. You need not worry about vertically aligning the matrix entries.

Sample input 1

```
4 2
13 99
64 43
95 65
2  86

2  10
31  3 67 50 78 20 63 27  9 83
12 28 12  1 97 98 57 47  9 54

10 3
87 29 31
49 86 60
70 16 23
15  2 37
46 75 74
56 80 72
12 54 50
93 73 65
41 26 14
78 17 79
```

Sample output 1

```
2563169 2765235 2966020
2461379 2181226 2649776
3680395 3267905 3964780
2011906 2246030 2359720
```